Information Processing Letters 116 (2016) 419–422

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A randomized algorithm for long directed cycle ☆

Meirav Zehavi

Department of Computer Science, Tel Aviv University, Israel

ARTICLE INFO

Article history:
Received 9 November 2015
Received in revised form 20 February 2016
Accepted 20 February 2016
Available online 24 February 2016
Communicated by B. Doerr

Keywords:
Algorithms
Parameterized complexity
Long directed cycle
k-Path

ABSTRACT

Given a directed graph G and a parameter k, the Long Directed Cycle (LDC) problem asks whether G contains a simple cycle on at least k vertices, while the k-Path problem asks whether G contains a simple path on exactly k vertices. Given a deterministic (randomized) algorithm for k-Path as a black box, which runs in time $t(G,k)$, we prove that LDC can be solved in deterministic time $O^*(\max\{t(G,2k),4^{k+o(k)}\})$ or in randomized time $O^*(\max\{t(G,2k),4^k\})$. In particular, we get that LDC can be solved in randomized time $O^*(4^k)$.

1. Introduction

We study the Long Directed Cycle (LDC) problem. Given a directed graph $G=(V,E)$ and a parameter k, it asks whether G contains a simple cycle on *at least* k vertices. At first glance, this problem seems quite different from the well-known k-Path problem, which asks whether G contains a simple path on *exactly* k vertices: while k-Path seeks a solution whose size is exactly k, the size of a solution to LDC can be as large as $|V|$. Indeed, in the context of LDC, Fomin et al. [1] note that "color-coding, and other techniques applicable to k-Path do not seem to work here."

In this paper, we show that an algorithm for k-Path can be used as a *black box* to solve LDC efficiently. More precisely, suppose that we are given a deterministic (randomized) algorithm PathAlg that uses $t(G,k)$ time and $s(G,k)$ space, and decides whether G contains a simple path on *exactly* k vertices directed from v to u for some given vertices $v,u\in V$.[1] Then, we prove that LDC can be solved in deterministic time $O^*(\max\{t(G,2k),4^{k+o(k)}\})$ and $O^*(\max\{s(G,k),4^{k+o(k)}\})$ space (if PathAlg is deterministic), or in randomized time $O^*(\max\{t(G,2k),4^k\})$ and $O^*(s(G,k))$ space (if PathAlg is randomized).[2] Somewhat surprisingly, we show that cases that cannot be efficiently handled by calling an algorithm for k-Path, can be efficiently handled by merely using a combination of a simple partitioning step and Breadth-First Search (BFS).

The first parameterized algorithm for LDC, due to Gabow and Nie [2], runs in time $O^*(k^{O(k)})$. Then, Fomin et al. [1] gave a deterministic parameterized algorithm for LDC that runs in time $O^*(8^{k+o(k)})$ using exponential space. Recently, Fomin et al. [3] and Shachnai et al. [4] modified the algorithm in [1] to run in deterministic time $O^*(6.75^{k+o(k)})$ using exponential space. These algorithms are also presented in the new monograph [5]. It is known that k-Path can be solved in randomized time $O^*(2^k)$ and polynomial space [6], and deterministic time $O^*(2.59606^k)$ and exponential space [7]. Thus, we immediately obtain that LDC can be solved in randomized time $O^*(4^k)$ and polynomial space, and deterministic time $O^*(6.73953^k)$ and exponential space. We briefly mention that the undirected variant of LDC is the special case of LDC where

☆ *Abbreviations*: Long Directed Cycle (LDC).
E-mail address: meizeh@post.tau.ac.il.

1 Known algorithms for k-Path handle the condition relating to the vertices v and u.

2 The O^* notation hides factors polynomial in the input size.

http://dx.doi.org/10.1016/j.ipl.2016.02.005
0020-0190/

Algorithm 1 PolyAlg($G=(V,E),k,L,R$).

```
1: for all v ∈ L and u ∈ L \ {v} do
2:     Use BFS to find a simple path P = (V_P, E_P) from v to u in G[L]
       that minimizes |V_P|.
3:     if |V_P| ≠ k or the path P does not exist then
3:         Skip the rest of this iteration.
4:     end if
5:     Use BFS to find a simple path P' = (V'_P, E'_P) from u to v in G[V \
       (V_P \ {v, u})] that minimizes |V'_P|.
6:     if the path P' exists then
7:         Accept.
8:     end if
9: end for
10: Reject.
```

for every edge $(v,u)\in E$, it holds that $(u,v)\in E$. Gabow and Nie [2] note that "the directed case is believed to be harder". Indeed, the first parameterized algorithm for the undirected variant has already been given in [8]. Further information can be found in [2].

In the following sections, given a graph $G=(V,E)$ and a set $U\subseteq V$, we let $G[U]$ denote the subgraph of G induced by U.

2. Finding large partitioned solutions in polynomial time

We say that an instance (G,k) of LDC *seems difficult* if G does not contain a directed cycle on ℓ vertices for any $\ell\in\{k,k+1,\dots,2k\}$. Roughly speaking, given such an instance, we are forced to determine whether G contains a *large* solution. This case, as noted in [2] and [1], seems to be the core of difficulty of LDC. We show, somewhat surprisingly, that under certain conditions, this case can be solved in polynomial time. More precisely, this section proves the correctness of the following lemma.

Lemma 1. *Let (G,k) be an instance of* LDC*, and let (L,R) be a partition of V. Then, there is a deterministic polynomial-time algorithm,* PolyAlg*, which satisfies the following conditions.*[3]

- *If (G,k) seems difficult, and G contains a simple cycle $v_1\to v_2\to\dots\to v_t\to v_1$ such that $t>2k$, $v_1,v_2,\dots,v_k\in L$ and $v_{k+1},v_{k+2},\dots,v_{2k}\in R$,* PolyAlg *accepts.*
- *If G does not contain a simple cycle on at least k vertices,* PolyAlg *rejects.*

Proof. The pseudocode of PolyAlg is given in Algorithm 1. Clearly, if the algorithm accepts, there exist two distinct vertices v and u such that G contains two simple internally vertex disjoint paths, $P=(V_P,E_P)$ (from v to u) and $P'=(V'_P,E'_P)$ (from u to v), where $|V_P|=k$. In this case, G contains a simple cycle, which consists of these paths, on at least k vertices. Thus, the second item is correct.

Now, we turn to prove the first item. To this end, suppose that the condition of this item is true. Then, we let $C=v_1\to v_2\to\dots\to v_t\to v_1$ be a simple cycle in G such that $t>2k$, $v_1,v_2,\dots,v_k\in L$ and $v_{k+1},v_{k+2},\dots,v_{2k}\in R$, *which minimizes t*. We need the following observations.

Observation 1. *The number of vertices on the shortest path from v_1 to v_k in $G[L]$ is exactly k.*

Proof. We let $P=(V_P,E_P)$ denote a path from v_1 to v_k in $G[L]$ that minimizes $|V_P|$. By the existence of C, such a path P exists and satisfies $|V_P|\le k$. Since $V_P\subseteq L$ and $v_{k+1},v_{k+2},\dots,v_{2k}\in R$, it holds that $v_{k+1},v_{k+2},\dots,v_{2k}\notin V_P$. Now, consider the (not necessarily simple) path $\widetilde{P}=(V_{\widetilde{P}},E_{\widetilde{P}})$ from v_{2k+1} to v_k obtained by concatenating P to the simple path from v_{2k+1} to v_1 that is a subpath of C. Observe that $|V_{\widetilde{P}}|=(t-2k)+|V_P|$ and that $v_{k+1},v_{k+2},\dots,v_{2k}\notin V_{\widetilde{P}}$. Thus, there exists a *simple* path from v_{2k+1} to v_k on at most $(t-2k)+|V_P|$ vertices that avoids $v_{k+1},v_{k+2},\dots,v_{2k}$. Together with the simple subpath from v_k to v_{2k+1} of C, we obtain a simple cycle on $(t-2k)+|V_P|+k=t-k+|V_P|$ vertices. If $|V_P|<k$, this cycle contains less than t vertices (but more than $2k$ vertices, since it contains $v_{k+1},v_{k+2},\dots,v_{2k}$ and (G,k) seems difficult), contradicting the choice of C. Thus, we conclude that $|V_P|=k$. □

Observation 2. *Let $P=(V_P,E_P)$ be a simple path from v_1 to v_k in $G[L]$ such that $|V_P|=k$. Then, $G[V\setminus(V_P\setminus\{v_1,v_k\})]$ contains a path from v_k to v_1.*

Proof. If $V_P\cap\{v_{k+1},v_{k+2},\dots,v_t\}=\emptyset$, the claim is clearly true, since then $v_k\to v_{k+1}\to\dots\to v_t\to v_1$ is a path in $G[V\setminus(V_P\setminus\{v_1,v_k\})]$. Suppose, by way of contradiction, that $V_P\cap\{v_{k+1},v_{k+2},\dots,v_t\}\neq\emptyset$. Since $V_P\subseteq L$ and $v_{k+1},v_{k+2},\dots,v_{2k}\in R$, it holds that $v_{k+1},v_{k+2},\dots,v_{2k}\notin V_P$ and $V_P\cap\{v_{2k+1},v_{k+2},\dots,v_t\}\neq\emptyset$. Now, consider the *non-simple* path $\widetilde{P}=(V_{\widetilde{P}},E_{\widetilde{P}})$ from v_{2k+1} to v_k obtained by concatenating P to the simple path from v_{2k+1} to v_1 that is a subpath of C. Observe that $|V_{\widetilde{P}}|=t-k$ and that $v_{k+1},v_{k+2},\dots,v_{2k}\notin V_{\widetilde{P}}$. Thus, there exists a *simple* path from v_{2k+1} to v_k on at most $t-k-1$ vertices that avoids $v_{k+1},v_{k+2},\dots,v_{2k}$. Together with the simple subpath from v_k to v_{2k+1} of C, we obtain a simple cycle on at most $t-1$ vertices. This cycle contains less than t vertices (but more than $2k$ vertices, since it contains $v_{k+1},v_{k+2},\dots,v_{2k}$ and (G,k) seems difficult), contradicting the choice of C. □

Consider the iteration of Step 1 that corresponds to $v=v_1$ and $u=v_k$. The first observation implies that the condition of Step 3 is false. Next, the second observation implies that the condition of Step 6 is true, and therefore PolyAlg accepts. □

3. Computing the sets L and R

In this section we observe that the computation of the sets L and R can merely rely on a simple partitioning step that is based on color coding [9]. To this end, we need the following definition and known result.

Definition 1. Let $\mathcal{F}$ be a set of functions $f:\{1,2,\dots,n\}\to\{0,1\}$. We say that $\mathcal{F}$ is an (n,t)-universal set if, for every subset $I\subseteq\{1,2,\dots,n\}$ of size t and every function $f':I\to\{0,1\}$, there is a function $f\in\mathcal{F}$ such that, for all $i\in I$, $f(i)=f'(i)$.

[3] In cases not covered by these conditions, PolyAlg can either accept or reject.